Sink the fleet P61920_en

¿Who has never played to ships with the mate of your side during one of those programming contests in which nothing turns out? OIE, despite what might seem, is a serious competition: we are going to give you the opportunity to play to ships... against the computer.
In this problem your program will play to sink the fleet. Unlike other problems, the input that your program will receive will be *interactive*: your program will print a square where to attack, and it will receive an answer
In a sink the fleet game, there is a rectangular map of R by C squares where ships of different size are randomly distributed . A ship is a row or a column of t consecutive squares, where t is from 1 (coast guards) to 5 (aircraft carriers). Two squares of different ships will never be in touch sharing an edge they can be in touch sharing a point, that is, diagonally placed one respect the other one). Your program knows how many ships of each kind there are, but it does not know where are placed. Your aim is to sink all of them. To sink a ship is necessary to damage all its squares.
In each turn, your program has the right to do an attack over any of the squares, writing its coordenates. In return, it will receive one of the three possible answer in its input: `WATER` (if the square is empty), `TOUCHED` (if the square is occupied by a ship that still has none touched squares) or `SUNK` (if the square is occupied by a ship which squares have been touched). For instance, if you sink a ship of n squares with n attacks, your program will receive $n-1$ answers of `TOUCHED` and a answer of `SUNK`; if you attack again any of the squares of the same ship you will receive the message of `SUNK`.

Input

Various test boards exist, all of them with different sizes and number of ships. A test data start with the number N of boards.
For each board, you must read the numbers R and C in a line (the number of rows and columns of the board) followed by a second line with exactly 5 numbers: the quantities c_i of ships of size i, for i from 1 to 5. Since that moment the input that it receives will depend on your attacks. When your program has sunk all the ships, it must print `SUNK` in the output, and start to read the following board. When it has finished the number of boards N, your program must finish.

Output

Your program must print each attack in a line, with the coordenates i and j of the attack, separated by a space, being fulfilled that $0 \leq i < F$ and $0 \leq j < C$. Print `END`, also in a line, when you detect that there are not more ships.

Observation

You will be able to use a simulator and various instance boards to check yourself your program. We also offer you and instance of an extremely silly player (`tonto.cc`) in order to have a code instance. You can base on the intance to write your program. To try an executable `jug.x` with the boards that we give you, write `./prueba jug.x`. You will receive as answer the number of turns that your program takes to print `END` for each test data. If your program has printed `END` before sinking all the ships, or if it has printed a square out of the board, it will appear `WA` (wrong answer) in the simulator. Check that your program does not take more than 10 seconds to solve the test data.

Score:
The system of puntuation of this exercise is different to the others. Only one sending will be allowed. The score of the problem will not be known until the next day, because it not depends only on your program, but also of the other contestants. When you have your program ready, you must send it to the judge. You can only do a sending; ignore the answer that the on-line judge gives you.
It will receive more points who obtains more chiqui-points overcoming particular test data, of different sizes and number of ships. All the program will have to overcome the same test data. For each test data overcomed we will give you chiqui-points. A program that, during the execution of a test data, attacks a none existing square or prints `END` before time will not receive any chiqui-point for the test data where it fails. The program that takes more turns to sink the ships of a test data will receive a chiqui-point; the next one, two; and this way to the most efficient. In a event of a tie, all the drawn programs will receive the maximal number of chiqui-points that they could receive. For instance, if 3 programs draw in the last position, the 3 of all will receive 3 chiqui-points; the following one will receive 4 chiqui-points (if it is not drawn), etc.
In the end, the points of the problem will be distributed in the following way,

$$\text{puntos} = \left\lceil 100 \cdot \left(1 - \frac{i-1}{N}\right) \right\rceil,$$

where N is the number of solutions that are received and i is the position that occupies in the accumulated classification of chiqui-points; that is, the first classified will receive 100 points, and the other classifieds will receive points proportionally to their position. In a event of a tie, the same criterion than before will be applied to share the points. The classification and the score will published the next day.
Author: Javi Gómez

Problem information

Author: Omer Giménez
Translator: Carlos Molina

Generation: 2026-01-25T11:11:03.669Z